R E P O R T R E S U M E S

ED 003 815 64

INSTRUCTOR'S GUIDE FOR INSTRUCTIONAL PROGRAMMING PROCEDURES.

BY- DETERLINE, WILLIAM A.
DETERLINE, WILLIAM A., SAN ANTONIO, TEXAS

REPORT NUMBER NDEA-VIIB-479-2 PUB DATE 64
EDRS PRICE MF-$0.09 HC-$0.80 20P.

DESCRIPTORS- *TEACHING GUIDES, *PROGRAMED TEXTS, *PROGRAMED INSTRUCTION, TEACHING MACHINES, SAN ANTONIO, TEXAS

AN INSTRUCTOR'S GUIDE SUMMARIZED THE AUTHOR'S VIEW TOWARD PROGRAMED INSTRUCTION MATERIAL OBJECTIVES, SUBJECT MATTER CONTENT, AND TRYOUT DATA. A COPY OF THE POST-TEST FOR THIS PROGRAMED COURSE IS INCLUDED. RELATED DOCUMENTS ARE ED 003 814 AND ED 003 816. (LP)

ED003815

B-479 ≅
No2
OFFICIAL FILE COPY

INSTRUCTOR'S GUIDE

TO

INSTRUCTIONAL PROGRAMMING PROCEDURES

William A. Deterline

U. S. DEPARTMENT OF HEALTH, EDUCATION AND WELFARE
Office of Education
This document has been reproduced exactly as received from the person or organization originating it. Points of view or opinions stated do not necessarily represent official Office of Education position or policy.

INSTRUCTOR'S GUIDE

FOR

INSTRUCTIONAL PROGRAMMING PROCEDURES

by

William A. Deterline

INTRODUCTION

The program, Instructional Programming Procedures, was written because I was tired of training programmers! After participating in two workshops conducted by the American Institute for Research in Pittsburgh in 1961, serving as a consultant to several industrial firms, Government agencies and the U. S. Air Force during 1962, and teaching graduate education courses in programmed instruction at Trinity University in San Antonio during 1962 and 1963, all of which involved some degree of training in the procedures and principles of programming, I decided to program as much as I could of the actual training courses. The first drafts of portions of this program were written late in 1962, and the tryout and revision process began immediately. By the summer of 1963 the first nine chapters had been completed. They were tested in my classes and those conducted by Dr. Carlton Downing at Trinity University and Col. Gabriel D. Ofiesh at Our Lady of the Lake College in San Antonio. Additional informal tryouts were obtained from military and civil service personnel of the Air Training Command at Lackland Air Force Base. Thanks are due Lt. Col. Kenneth Johnson, Major William Strother and Major Marvin Roever of the Instructional Systems Division at Lackland Air Force Base for distributing copies to interested individuals.

During the late months of 1963 and early 1964 the tryout data were collected from over seventy-five students and revisions were made on the basis of student responses to frames and examinations, and student comments and suggestions were incorporated where appropriate. The final units of the program were written during the Summer and Fall of 1964 and the entire program was given a final field test on twenty students late in 1964. A final revision was then done on the basis of the field test data.

One measure by which programs are often evaluated is the error rate of the frames in the program. Ideally the error rate should be kept low--some say between 5% and 10%--which would mean that most students make very few wrong responses in the frames of the program. The error rate of this program falls below 10% and only a few frames have error rates above 15%. When revisions were made, most of the frames with error rates above 10% were rewritten or the sequences in which they appear were rewritten, but some frames, intentionally made difficult as "thought problems", were not revised, since the immediate feedback was assumed to be adequate as a means of correcting the errors. One rule that was followed, however, was that no sequence was allowed to stand without revision if its terminal (criterion) frame held an error rate above 10%.

TARGET POPULATION

The target population for which the program is appropriate is difficult to specify, and the user or evaluator will have to operate on as vague a set of specifications as I had to use in designing the programs. I wanted a program that would train programmers; but how do we identify the characteristics of the fairly heterogeneous population that might by choice or assignment encounter this program as programmer trainees? The program is designed to teach industrial training personnel who have some experience, either as technical writers or as instructors, and to teach public school teachers and administrators, and undergraduate and graduate

students in education, and others who for any reason might use this program. How do we define the characteristics of the student population? I didn't. I tried to write for the entire group and undoubtedly have missed the mark as far as many potential students are concerned. During the tryouts I tried to sample as many different kinds of members of the target population as possible, so that feedback from the tryouts include data based on individuals ranging from age 18 to 46, from zero to over 20 years teaching or technical writing experience. All tryout subjects were high school graduates and more than half were college graduates. Since we have very little data that allow us to select programmers, I can only recommend that the program be used for individuals with some experience and success in teaching and/or writing, who are probably B students or better in college, who have a great deal of patience and resistance to boredom, and who can express themselves easily and clearly in writing. Those criteria, I realize, are poorly defined and are not very specific. Until effective screening and predictive measures are available that let us select potentially effective programmers, and do so with some degree of success, I'm afraid that this is as specific as I can be.

OBJECTIVES

The objectives of each chapter are represented by the questions or problems that appear in the pre-and post-tests that accompany each chapter. There are two basic classes of objectives, verbal and performance. The verbal objectives include having the student describe and explain certain rules, techniques and procedures, and the performance objectives include the application of all of the rules, techniques and procedures to the actual development of a program, from the specification of objectives, through the frame writing to the editing and tryout and revision process. For the performance requirements, the student is to use the APPENDIX as his guide. In the APPENDIX, which is partly produced by the student as he proceeds through the program, appear the rules, samples, check lists and outlines that will help him to program. In some cases the index will refer the student to portions of the program proper for prototype and sample sequences.

The reviewer interested in evaluating the effectiveness of this program can follow several procedures; one of which consists of evaluating student performance on the chapter post tests; another involves using all or selected portions of the chapter post tests as a final examination; the most crucial procedure involves giving the student a topic to program. The latter technique is probably the most effective. The student's program can then be evaluated in two ways; by comparing it's characteristics with those specified in the chapter post tests and in the APPENDIX; by evaluating the effectiveness of the student's program in teaching other students.

One point that I should mention is this: the program is not conceived of as a teaching instrument designed to produce a finished, polished and qualified programmer. It is designed to teach certain basic procedures well enough to produce an embryo programmer, and to provide him with the references he needs in order to produce programs. When he completes this program the student is not done with it, since the APPENDIX then becomes his guide and the program becomes a set of sample or prototype sequences that he can refer to in programming his own materials.

The student, after completing the program, still needs something that the program doesn't give him, and which I don't know how to program: some editorial assistance and feedback from a more experienced programmer, who can react immediately to the work that the student does, diagnosing errors, and reinforcing appropriate frame writing behavior. Like any kind of creative skill, programming requires a great deal of practice, and instead of trial and error work in isolation, the beginning programmer, after completion of this program, should be given as much individual attention and guidance as possible and he should interact with other programmers in order to profit from their experience.

SUBJECT MATTER CONTENT

Yes indeed

The content of this program is extremely idiosyncratic, reflecting my own biases more than it does a concensus of the views of all programmers, although I hope that the views expressed encounter more agreement than disagreement. My experience with a great many programs and programmers have led to these views. Although the techniques of programming are changing rapidly from year to year, I feel that the procedures taught in this program will serve as an effective basic training that can then be modified as the field of programming changes.

The language of programming is a babble of many dialects, and one of the favorite pastimes of programmers, apparently, is the invention of new terms and descriptive labels for their procedures, products, rules, principles and concepts. I have indulged myself in this sport in the program, justifiably I hope, in developing a simple and usable set of labels. I have largely ignored some of the terms used by other programmers, generally where I have felt that the terms didn't justify their existence or where several different terms were available for the same kind of procedure or characteristic. In some cases I have used familiar terms, but have limited their meaning. The label, "copy frame", for example, is considered by many to refer to a Bad Thing. Granted, the use of copy frames could be overdone. The overprompting produced by an inappropriate sequence of copy frames is undesirable, and is probably the easiest way to write poor programs. But for some purposes, a copy frame is justified and in other cases unavoidable.

For the person who wants to examine content without working through the frames, or for the student who has completed the program and wants to review, the final section is appropriate, since it discusses in unprogrammed narrative form the objectives and content of the entire program.

STUDENT REACTIONS

On the whole, student reactions to this program have been quite favorable. Some have found it boring, some have found it extremely interesting and enjoyable, some have found it quite tiring, some have found it too hard, and others have complained that it is so simple that it insults their intelligence! Some of these differences reflect the differences in the student population. In one group tryout, for example, I discussed the entire program with the students after they had completed it. A great argument developed in the group about one of the chapters. Several of the students thought that it was too hard, too complex and a bit vague. Several others said that it was clear, interesting, and quite easy. And a

few said that they were bored to distraction and found it childishly simple. In revising that chapter I attempted to make it all things to all people: I tried to make it clearer and a little easier, while adding what I thought would be material with increased interest value. The next group of tryout students agreed, almost unanimously, that that chapter was now too long and much too cluttered with miscellaneous information. I re-revised that chapter, making it more like it had been for the earlier group, and range of opinions about it promptly re-appeared, but that is the way that chapter stands in this version.

Students with prior experience in programming will find many parts of this program very dull indeed. This is to be expected since the program is designed for students with no previous experience. Although the program does not tell the student this, an instructor using this program to train programmers should evaluate every pre-test, and if the student correctly answers almost all of the questions correctly, he should be told to skip that chapter, and to read instead the unprogrammed summary of that chapter that appears in the final review chapter. He should also be told to refer to the answer page for the post-test items and to read the frames that correspond to each question that he missed. This by-pass option should belong to the instructor in order to insure that a student does not inappropriately by-pass a chapter he needs. In general, student reactions to this type of by-pass technique have been quite favorable.

As I noted above, students' expressed evaluations have for the most part been favorable and enthusiastic. Responses of this sort, however, are important only if freely and honestly given and even then provide only a tentative evaluation of the motivational characteristics rather than the instructional effectiveness of the program. The effectiveness of a program must ultimately be determined by subjecting it to use by students. If the student learns, the program passes. If the student doesn't learn, the program fails. Unfortunately, from my point of view, some students will not learn adequately from the program because they are not appropriate students, but that is unavoidable since I am unable specifically to identify the characteristics that would allow completely adequate selection of students. In that type of situation then, responsibility for any failures belongs to the program. And to me.

TRYOUT DATA

Following the next-to-last revision of the program, a final field test was conducted. Since most of the rough spots had already been removed in earlier tryouts, a relatively small student group was used for the final field test. Fifteen students took part in the field test. The students were college undergraduate and graduate students with no teaching or writing experience. None had previous experience or training in programming.

The procedures for the administration of the field test were as follows:

1. All students were volunteers who were paid a token amount ($5.00) to complete the program. In addition to the instructions contained in the program, the students were told to write all their responses in the blanks in the frames. They were also told to write comments about any frame or sequence that caused them difficulty, or to write comments for any other reasons.

2. Each student was requested to keep track of the amount of time actually spent on each chapter, including pre-test and post-test times.

3. He was told to keep all materials that he wrote (frames, objectives, etc.) and to turn them in with the completed program.

4. As each student completed the program he was given a written final examination that asked every verbal objective of the program and required a number of performance responses. The final examination is attached to this guide. Scores appear in Table 1.

5. The program produced by the student in the final unit of this program was then evaluated by inspection to determine the degree to which each student violated or followed the procedures and rules taught by this program. As expected, there were wide individual differences in writing ability and program quality was obviously related to this variable. The general level of performance was quite satisfactory, although the students obviously needed immediate editorial feedback. No attempt was made to provide numerical scores for the student programs.

The interaction between basic writing ability and principles learned from this program was the most interesting effect noted. In some cases, for example, it was apparent that a student was attempting to follow procedures he had learned, but simply did not have the verbal facility required to produce effective communication. Some students were unable, except in an "unimaginative manner" to write frames as specified by this program. Poorer students (as determined by college achievement) generally tended to write the poorest materials.

The data in Table 1 show the percentage of students answering each item correctly. The data are not complete for all students since the field test overlapped, unfortunately, with semester examinations and the press of time prevented some students from completing the entire program prior to the final deadline given them. Final revisions of the program and test were based on the data shown in Table 1 and on suggestions made by the tryout students.

The time required to complete the program cannot be predicted in advance. Some students methodically checked their answers to each unit post-test, while others did not. Some students spent many hours on the programming exercises, others devoted only minimal time and attention to the exercises. Because of the great variability possible in time required to complete the programming, editing, and tryout and revision exercises, the following time estimates are based only on the instructional portions of the program and do not include the independent programming assignments. The shortest time reported by a student was thirteen hours and fifty minutes, plus two and one-half hours for the post-test, a total of approximately sixteen and one-half hours. The longest time reported was thirty-five hours, plus three hours for the post-test, a total of approximately thirty-eight hours. Average time for the practice exercises should probably run between three and six hours. Considering the entire program, including all exercises, tests, and the final post-test, it can be estimated that completion time for the entire program will range from about twenty hours to as much as forty or fifty hours.

RECOMMENDATIONS

The program, because it requires continual active participation, is apparently quite fatiguing. I doubt that the program could be used effectively on an eight, or even six, hours a day basis. It is difficult to recommend any fixed scheduling procedure, since the context in which the program is to be used will largely determine the most effective method of implementation. In general, however, I would suggest that a student complete two units each day, or up to four hours of program use, whichever is greater. The student should at least examine each chapter pre-test, should take and score each post-test, read the narrative reviews for at least the last completed chapter before beginning a new chapter and should complete each chapter in a single sitting. That is, once a chapter has been started, he should not stop until that chapter has been completed. The reason for this last suggestion is obvious: if students select their own stopping point, they will almost certainly stop in the middle of a sequence that does not lend itself well to being broken.

If the program is to be part of a workshop course of instruction, the program can be used for scheduled homework assignments, or in a manner that I will call pre-scheduled self-pacing. By this, I mean that the materials will be scheduled by chapters and days so that the class progress is in effect scheduled, even though each student moves at his own pace through the scheduled assignments. For convenience and some degree of control, the last two or three hours of each day can be scheduled for program use in the classroom. The instructor should be available to provide individual assistance when necessary. Students who are unable to complete the day's assignment during the scheduled period are told to finish the assignment as homework. At the next class-meeting the instructor should begin by answering any questions asked by the students. He can then provide a test, similar to the chapter post-tests, for additional diagnostic purposes and to be sure that all students have in fact completed the assignment. Workshop exercises, lectures, group demonstrations and other activities can be scheduled for the rest of the day up to the scheduled period of program use.

I recommend strongly that each student be given considerable editing support and feedback, and constant supervision as he carries out the

assigned practice exercises in the final two chapters of the program. At that point he will be preparing objectives, writing short programs, editing them, trying them out, and revising them. He should have a skilled programmer available for assistance and advice. I could not, obviously, foresee all of the possible assists that the student might need while working on his own, and although I attempted to provide for some of the most likely difficulties, I knew that a completely adaptive trouble-shooting program was not possible.

There were many afterthoughts that occured to me during the final revision process, afterthoughts concerning content that I would like to have put into the objectives of the program in the first place. Several of these were concerned with ways in which the student might modify some of the specific techniques while still applying the basic principles he had learned. For example, a point that was so obvious that I forgot to include it, is that the immediate application type of frame can, for certain types of students and certain types of subject matters, be used for many kinds of objectives beyond the limted possibilities taught in this program. A frame that explains a point briefly and then asks a short direct question that requires either the covert or overt application of the information has many advantages -- if it works. Frames of that sort can be written rapidly without quite as much concern for specific sentence structures, and brighter students often prefer and learn as effectively from frames that interact more in a tutorial way with them. Afterthoughts of this nature, however, will have to wait for the next revised and expanded version of this program.

William A. Deterline

INSTRUCTIONAL PROGRAMMING PROCEDURES

Post Test

1. What characteristic of programmed instruction:
   a. allows a program to adjust to varying needs of students?

   b. insures that the students "pay attention"?

   c. helps the student remember?

   d. is a characteristic of format, or presentation, that ~~facilitates learning~~ makes learning easier?

2. What do we call the instructional goals of a program, and how are they always stated?

3. Name and describe briefly three kinds of information frames.

4. An information frame does what two things?

5. What kind of frame:
   a. must follow an information frame?

   b. should be used to introduce an unfamiliar new word?

   c. does not contain ~~the~~ new information needed for the response (3 answers)?

   d. requires a complete answer?

   e. uses an existing association?

   f. introduces a familiar new word?

   g. ~~introduces an unfamiliar new word?~~

6. What are the characteristics of a good program in terms of the amount of help given the student and the length and completeness of the student's response?

7. Name the two critical elements in every frame.

8. Into what two classes may the words or terms of a response be divided, and what is the basic difference between the two classifications?

9. Mark with an "X" those of the following frames that have adequate stimulus content if they were to appear on the last page of the program.
   a. A criterion frame is * _____.
   b. Adaptive sequencing * _____.
   c. Every frame requires a(n) _____ response.
   d. The "90-90" criterion means * _____.
   e. You use a copy frame when * _____.

10. By what sequence design do you teach a term if:
   a. it is specific?

   b. it is general?

   c. you can't decide whether it's specific or general?

11. What is the first thing you decide before beginning to write a frame? Why?

12. Underline all the terms in this response that are specific?

> A thematic prompt, in some programming terminologies, is the use of an existing association to cue a response.

13. One method of programming, which ~~that~~ uses the branching technique of adaptive sequencing exclusively, is called "intrinsic" programming.
   a. Write a copy frame for the word "intrinsic" in this context. (Your target population has just completed this program)

   b. Write a practice frame for "intrinsic".

   c. Write a cue frame for the word "branching" in this context.

   d. Write a combination frame for the response "branching", and for "adaptive sequencing", a new response.

e. Write a criterion frame for the definition of "intrinsic".

14. If we teach:

A prefix that means "between" is inter

can we expect the student to respond correctly to:

The prefix "inter" means between?

(Answer yes or no, and explain)

15. What do we call a diagram or data table that helps the student respond? What do we call a memorized aid of this type?

16. Why should you "bury" the response to be copied in a copy frame, rather than underlining or capitalizing it?

17. List the three types of adaptive sequencing and describe the characteristics of each.

18. Describe a copy frame -- how many sentences, or parts, does it have and what is in each?

19. What are the possible combinations of rules and examples in a frame under the RULEX system?

20. Describe the relationship, if any, between the stimulus elements in the parts of a copy frame.

21. The four types of existing associations used in cue frames are classifications, opposites, obvious conclusion and related terms. Write one example frame for each of these types of association.

22. What kind of test samples the lessons objectives?

23. What kind of test measures every objective of the lesson?

24. What are the two characteristics of a discrimination question?

25. Define "Constructed Verbal Response".

26. Mark the following response objectives CVR, DISC, VPERF (verbal performance) or MPERF (non-verbal performance).

    a. Give a speech.
    b. Identify a frame type.
    c. Write a copy frame.
    d. Load a camera.
    e. Define "intrinsic".
    f. Write a program.

27. ~~How should you teach an objective, regardless of the response-type label you put on it? (How do you decide what class response to call for?)~~
How must every specific term be introduced, and why do we have this rule?

28. What kind of information frame usually begins a discrimination sequence?

29. Here is a discrimination objective. Write the first two frames of the teaching sequence.

When the form of a verb that ends in "ing" acts as a noun in a sentence, it is called a gerund. (Playing the piano is an art.) Student will discriminate gerunds from other "ing" words used in sentences.

30. How can you provide confirmation in a performance sequence (2 ways)?

31. List and describe the five steps of preparing objectives for programming.

32. If a student learns a sequence of responses, what becomes the stimulus for each succeeding response?

33. What is the purpose of the criterion test?

34. Criterion test questions generally take one of what two forms?

35. What relationship is there between the response-class of an objective, the sequence that teaches it, and the criterion test question that measures it?

36. What do you evaluate in the first several steps of editing? With whom do you confer?

37. What is the general rule about the spacing of review frames?

38. What must you remember when inserting review frames into a programmed sequence?

39. Name two types of review techniques.

40. What information should be published with the program for the benefit of the purchaser?

41. How does a programmer find out whether his program ~~is adequate?~~ accomplishes its objectives?

42. How much are we sure the student will learn and remember of discussion material?

43. Mathetics is a programmed instruction method that combines some of the characteristics of both the linear and branching methods. One characteristic of mathetics, called "chaining", is that procedures are taught backwards! For example, if you were using a mathetic approach to teach someone to load a camera, you would teach the last step first, putting the film in position for the first picture. Then you would teach the step preceding the last step and review the last step. Your student would close the camera and adjust the film for the first picture. Next he would attach the new film to the empty spool, close the camera, and so on. The theory behind this "backwards" approach is that seeing the results of the finished procedure acts as confirmation and reinforcement (a reward) for the learner.

You are going to teach the above material on mathetics. Prepare your objectives following the five steps, and write the frames. Use additional blank paper for your program.